We Claim:

1. In a modeling and execution environment, a method comprising the steps of:

providing a graphical debugger interfaced with a model view of a model being executed, said graphical debugger having debug information related to the execution of said model, said debug information indicating at least one of the order of the execution of a plurality of methods in said model and a start time and a stop time of at least one method executed during the execution of said model; and

outputting said debug information to a user.

2. The method of claim 1, comprising the further steps of:

wrapping data generated by the execution of said model in an object, said wrapping encapsulating said execution-generated data in said object; and

exposing said data to said debugger via at least one interface to said object.

3. The method of claim 2, comprising the further step of:

altering said data via said interface.

4 . The method of claim 2 wherein said execution-generated data is at least one of state information, block inputs, block outputs, solver data, profiling data and signal values for said model.

5. The method of claim 1, comprising the further steps of:

processing said model to create compiled model information; and

programmatically generating executable code from said compiled model information, said code including an interface to said debugger.

6. The method of claim 5, comprising the further step of:

executing said generated code wherein said debugger at least one of sends and receives information from said executing code during said execution.

7. The method of claim 6, comprising the further steps of:

saving the execution history of said executable code; and

outputting the execution history by at least one of saving it in a permanent memory location, displaying it for a user, and sending it to a printing device to be printed.

8. The method of claim 6 wherein said debugger is started after compilation and before the execution of said code.

9. The method of claim 1, comprising the further step of:

indicating graphically with said debugger a plurality of blocks that are part of an algebraic loop when execution of the model is processing the algebraic loop.

10. The method of claim 1, comprising the further step of:

saving a record of a unique method invocation, said unique method invocation being information related to the execution of a method that belongs to at least one of a block, system, and model instance in an execution list of called methods.

11. The method of claim 10 wherein said unique method invocation record includes information regarding child records of methods executed inside the unique method invocation record.

12. The method of claim 11 wherein a link is provided from said unique method invocation record to said child record.

13. The method of claim 10 wherein said unique method invocation record includes information regarding at least one parent record of methods in which the unique method invocation is executed.

14. The method of claim 13 wherein a link is provided from said unique method invocation record to said parent record.

15. The method of claim10 wherein said unique method invocation record includes data about a state of the method invocation.

16. The method of claim 15 wherein said state indicates the method invocation is at one of the states of entering, entered, exiting and exited.

17. The method of claim 1, comprising the further step of:

communicating with an external mode simulation with said debugger.

18. The method of claim 1, comprising the further step of:

saving a snapshot of data relating to model execution during execution, said snapshot data being sufficient to enable the subsequent restarting of the execution of said model using said snapshot data at a saved point in time.

19. The method of claim 18 wherein said snapshot data is saved programmatically at at least one of a regular interval or based on a user-defined parameter.

20. The method of claim 19, comprising the further step of:

loading a saved snapshot into said debugger; and

executing the saved model from the point in time said snapshot was saved.

21. The method of claim 18, comprising the further step of:

displaying graphically to a user the saved snapshot data.

22. The method of claim 21, comprising the further step of:

displaying graphically to a user at least one additional set of snapshot data without restarting the execution of said model.

23. The method of claim 22 wherein said snapshots are displayed in order of decreasing time.

24. The method of claim 18, comprising the further step of:

saving a difference between a set of current model execution data and a saved snapshot.

25. A medium for use in a modeling and execution environment on an electronic device, said medium holding executable instructions on the electronic device for performing a method, said method comprising the steps of:

providing a graphical debugger interfaced with a model view of a model being executed, said graphical debugger having debug information related to the execution of said model, said debug information indicating at least one of the order of the execution of a plurality of methods in said model and a start time and a stop time of at least one method executed during the execution of said model; and

outputting said debug information to a user.

26. The medium of claim 25, wherein said method comprises the further steps of:

wrapping data generated by the execution of said model in an object, said wrapping encapsulating said execution-generated data in said object; and

exposing said data to said debugger via at least one interface to said object.

27. The method of claim 26, comprising the further step of:

altering said data via said interface.

28. The medium of claim 26 wherein said execution-generated data is at least one of state information, block inputs and outputs and signal values for said model.

29. The medium of claim 25, wherein said method comprises the further steps of:

processing said model to create compiled model information; and

programmatically generating executable code from said compiled model information, said code including an interface to said debugger.

30. The medium of claim 29, wherein said method comprises the further step of:

executing said generated code wherein said debugger sends and receives information from said executing code during said execution.

31. The medium of claim 30 wherein said method comprises the further steps of:

saving the execution history of said executable code; and

outputting the execution history by at least one of saving it in a permanent memory location, displaying it for a user, and sending it to a printing device to be printed.

32. The medium of claim 30 wherein said debugger is started following the initiation of the execution of said code.

33. The medium of claim 25, wherein said method comprises the further step of:

indicating graphically with said debugger a plurality of blocks that are part of an algebraic loop when execution of the model is processing the algebraic loop.

34. The medium of claim 25, wherein said method comprises the further step of:

saving a record of a unique method invocation, said unique method invocation being information related to the execution of a method that belongs to at least one of a block, system, and model instance in an execution list of called methods.

35. The medium of claim 34 wherein said unique method invocation record includes information regarding child records of methods executed inside the unique method invocation record.

36. The medium of claim 35 wherein a link is provided from said unique method invocation record to said child record.

37. The medium of claim 34 wherein said unique method invocation record includes information regarding at least one parent record of methods in which the unique method invocation is executed.

38. The medium of claim 37 wherein a link is provided from said unique method invocation record to said parent record.

39. The medium of claim 34 wherein said unique method invocation record includes data about the state of the method invocation.

40. The medium of claim 39 wherein said state indicates the method invocation is at one of the states of entering, entered, exiting and exited.

41. The medium of claim 25, wherein said method comprises the further step of: communicating with an external mode simulation with said debugger.

42. The medium of claim 25, wherein said method comprises the further step of:

saving a snapshot of data relating to model execution during execution, said snapshot data being sufficient to enable the subsequent restarting of the execution of said model using said snapshot data at a saved point in time.

43. The medium of claim 42 wherein said snapshot data is saved programmatically at at least one of a regular or user-defined interval.

44. The medium of claim 42, wherein said method comprises the further step of:

loading a saved snapshot into said debugger; and

executing the saved model from the point in time said snapshot was saved.

45. The medium of claim 42 wherein said method comprises the further step of:

displaying graphically to a user the saved snapshot data.

46. The medium of claim 45 wherein said method comprises the further step of:

displaying graphically to a user at least one additional set of snapshot data without restarting the execution of said model.

47. The medium of claim 46 wherein said snapshots are displayed in order of decreasing time.